Quality of Experience-Centric Management of Adaptive Video Streaming Services: Status and Challenges

STEFANO PETRANGELI, JEROEN VAN DER HOOFT, TIM WAUTERS, and FILIP DE TURCK, Ghent University - imec, IDLab

Video streaming applications currently dominate Internet traffic. Particularly, HTTP Adaptive Streaming (HAS) has emerged as the dominant standard for streaming videos over the best-effort Internet, thanks to its capability of matching the video quality to the available network resources. In HAS, the video client is equipped with a heuristic that dynamically decides the most suitable quality to stream the content, based on information such as the perceived network bandwidth or the video player buffer status. The goal of this heuristic is to optimize the quality as perceived by the user, the so-called Quality of Experience (QoE). Despite the many advantages brought by the adaptive streaming principle, optimizing users' QoE is far from trivial. Current heuristics are still suboptimal when sudden bandwidth drops occur, especially in wireless environments, thus leading to freezes in the video playout, the main factor influencing users' QoE. This issue is aggravated in case of live events, where the player buffer has to be kept as small as possible in order to reduce the playout delay between the user and the live signal. In light of the above, in recent years, several works have been proposed with the aim of extending the classical purely client-based structure of adaptive video streaming, in order to fully optimize users' QoE. In this paper, a survey is presented of research works on this topic together with a classification based on where the optimization takes place. This classification goes beyond client-based heuristics to investigate the usage of server- and network-assisted architectures and of new application and transport layer protocols. In addition, we outline the major challenges currently arising in the field of multimedia delivery, which are going to be of extreme relevance in future years.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Information systems** → **Multimedia streaming**; • **Networks** → Network management; Public Internet;

Additional Key Words and Phrases: Quality of Experience, HTTP Adaptive Streaming, MPEG-DASH

ACM Reference format:
Stefano Petrangeli, Jeroen van der Hooft, Tim Wauters, and Filip De Turck. 0000. Quality of Experience-Centric Management of Adaptive Video Streaming Services: Status and Challenges. *ACM Trans. Multimedia Comput. Commun. Appl.* 0, 0, Article 00 (0000), 28 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Internet traffic is currently dominated by video streaming applications. Video traffic is expected to grow from 42 Exabytes per month in 2016 to 159 in 2021, an impressive 279% growth rate [16]. One of the success factors for this diffusion is the wide adoption of the HTTP adaptive streaming (HAS) principle, which has gradually replaced traditional delivery techniques using the RTP/RTSP

Author's addresses: Stefano Petrangeli, Jeroen van der Hooft, Tim Wauters, Filip De Turck: Department of Information Technology, Technologiepark-Zwijnaarde 15, B-9052 Ghent, Belgium. Email: first.last@ugent.be.

1551-6857/0000/00-ART00 $15.00
https://doi.org/0000001.0000001

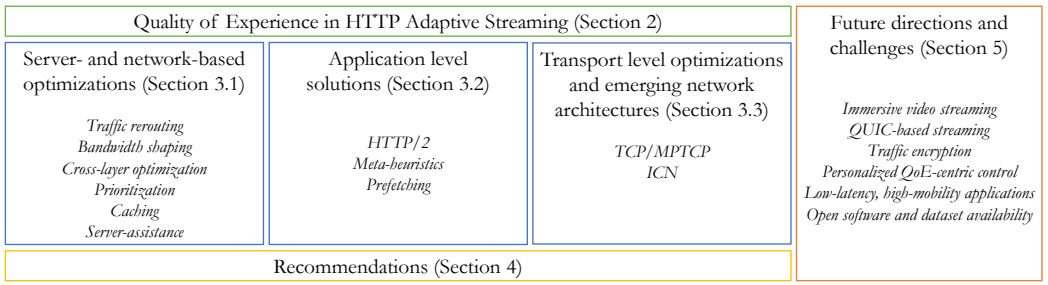


Fig. 1. Starting from QoE aspects in HAS, we then review the status of QoE-centric management solutions and present future research directions and challenges.

protocol suite and progressive download. In HAS, the video is encoded at different quality levels and temporally segmented, so that each segment is an independent object or file. The client is informed of the characteristics of the video via a Media Presentation Description (MPD), which describes the available bitrates and quality levels, among others. A rate adaptation heuristic, deployed at the client, dynamically decides the bitrate of each segment to download, based on the buffer status and the perceived network conditions. The goal of the heuristic is to match the video bitrate to the network bandwidth, with the primary focus of providing a continuous playout while maximizing the streamed quality. Being based on the HTTP protocol, this approach allows for an easy deployment and firewall traversing.

In light of the wide adoption of the HAS principle in many proprietary solutions, the MPEG consortium has created a standard for HAS in 2012, called Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [89]. Standardization mainly covers the structure of the MPD of the video, while the client rate adaptation heuristic is outside MPEG-DASH's scope. Moreover, MPEG-DASH is completely codec-agnostic, meaning it is compatible with any codec format.

Since the standard was defined, a large body of research has investigated how to improve users' Quality of Experience (QoE) for HAS [50, 84, 87]. Many studies focus on the development of new heuristics, in order to improve the quality adaptation of the client [41, 57, 93]. In recent years though, several approaches have been proposed that go beyond traditional client-based algorithms, with the goal of fully optimizing users' QoE [19, 78, 107]. This shift was needed as several QoE aspects in HAS cannot be completely optimized by the heuristic itself. For instance, despite the capability to adapt to varying network conditions, current HAS solutions can still suffer from video freezes. The 2015 Conviva report shows that almost 25% of the analyzed HAS sessions are affected by at least one freeze [21]. The same report also highlights that low video quality still impacts 54% of the sessions. This problem is mainly due to the unmanaged nature of current HAS technologies, as the clients are only aware of the local perceived bandwidth conditions and cannot be assisted in improving the delivered QoE. Moreover, the occurrence of freezes becomes more prominent during live streaming sessions, where the video player buffer has to be reduced as much as possible in order to minimize the live latency. In current HAS deployments, this latency is in the order of tens of seconds, because a large buffer at the client is usually required to minimize playout freezes and the server only sends a new video segment once a request is issued by the client. A purely client-based solution can also perform sub-optimally when multiple streaming clients compete for shared bandwidth [5]. Fairness issues are not due to TCP dynamics, but mainly arise from the rate adaptation algorithms, as they decide on the actual rate to download. When multiple clients retrieve video at the same time, wrong bandwidth estimations can occur, due to the temporal overlap of the activity-inactivity periods of different clients. This wrong estimation subsequently affects the bitrate selection and thus the final QoE. Previous examples identify scenarios where

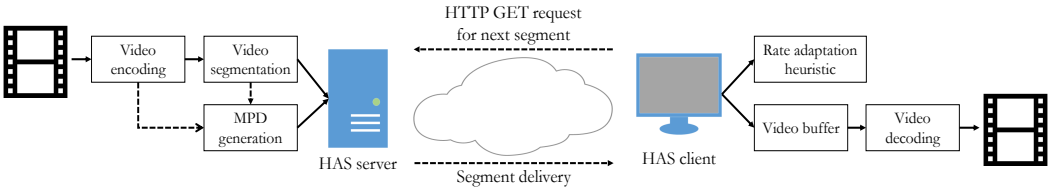


Fig. 2. In HAS, a server hosts the video, which is encoded at different qualities and segmented. A rate adaptation heuristic, deployed at the client, determines the quality level to be downloaded.

purely client-based solutions are not able to guarantee the best QoE to the users. Consequently, this survey focuses on existing works that optimize the delivery architecture of classic HAS systems, beyond client-based heuristics. These approaches can be categorized into three groups, based on where the optimization takes place: (1) server- and network-based approaches, (2) application level optimizations and (3) transport- and network-layer modifications.

The contributions of this paper are twofold. First, we provide an overview on the *status* of existing works in the aforementioned areas and provide guidelines on the best strategy to adopt depending on the QoE factor to be optimized and the deployment complexity. Second, we identify six main open *challenges* that will need to be addressed by the multimedia delivery community in future years, namely: (1) immersive video streaming, (2) QUIC-based adaptive streaming, (3) video traffic encryption and analysis, (4) personalized QoE modeling and control, (5) video delivery for low-latency, high-mobility applications and (6) open software, testbeds and datasets. The analysis presented in this work complements previous surveys on HAS, which mostly focus on existing rate adaptation heuristics only [50, 84, 87]. Moreover, this paper goes one step further by giving a structured outline of QoE-centric solutions and an outlook on future directions. In light of the above, this paper can help fostering future research in the adaptive video streaming domain.

The remainder of this paper is structured as depicted in Figure 1. Section 2 introduces the main elements of the HAS principle. Particularly, Section 2.1 describes the general architecture of adaptive streaming services, while Section 2.2 describes the main factors influencing QoE in HAS. Section 3 reports the current status of QoE-centric management of HAS services, reviewing existing works and clustering them in three different groups: server- and network-based solutions (Section 3.1), application level optimizations (Section 3.2) and transport level modifications (Section 3.3). In order to facilitate future research in this domain, Section 4 provides general guidelines on the best solution to adopt based on the network and streaming scenario. Future challenges for the multimedia delivery community are discussed in Section 5. Section 6 concludes the paper.

2 THE HTTP ADAPTIVE STREAMING PRINCIPLE

2.1 Architecture and Components

A traditional HAS architecture is composed of two elements: a server hosting the MPD of the video and the content itself, and a client streaming the video using the HTTP protocol (Figure 2). At the server-side, the content is encoded at different quality levels and temporally segmented, each segment usually being 1 to 10 seconds long, depending on the deployment. Each quality level can be decoded independently from the others and is characterized by a specific encoding bitrate, resolution, framerate and codec [106]. When layered encoding is used instead, higher quality levels can only be decoded in combination with lower layers [85]. This approach reduces the storage requirements for the video at the cost of an increased encoding overhead per layer. In order to start a streaming session, the client first downloads and parses the MPD of the video to retrieve information on the available quality levels. A rate adaptation heuristic dynamically decides the best quality level to download, for example based on network conditions and device capabilities.

Once the right quality has been selected, an HTTP GET request is issued to the server to download the segment. This selection is repeated periodically, after each segment has been downloaded. The client is also equipped with a buffer where segments are stored before being played, which is used to absorb temporary bandwidth fluctuations and avoid video freezes as much as possible.

Adaptation heuristics can be roughly divided into three groups: throughput-based, buffer-based and hybrid. Throughput-based heuristics mostly rely on the estimation of the available bandwidth performed by the client to select the bitrate of the next segment. The CS2P algorithm by Sun et al. falls in this category [93], for example. The authors improve the throughout prediction of HAS clients by developing a prediction model based on past video sessions, which is built offline in a node located in the streaming service provider network. This model is then used online by the clients, which remain the sole responsible of the actual quality adaptation. Buffer-based heuristics only use the buffer filling level to make a decision on the quality to request. The BBA heuristic from Huang et al. defines three operational regions based on the buffer occupancy B: reservoir ($B < B_{min}$), upper reservoir ($B > B_{max}$) and cushion ($B_{min} < B < B_{max}$) [41]. In the reservoir and upper reservoir regions, the client requests the lowest and highest bitrate, respectively. In the cushion region, a monotonically increasing function is defined to select the bitrate based on the buffer. Another buffer-based heuristic is the BOLA algorithm proposed by Spiteri et al. [91]. The authors formulate the adaptation process as a utility maximization problem, where the utility depends on the achieved quality and number of video freezes. An online control algorithm is developed using the Lyapunov optimization, which guarantees the achieved utility is within a certain limit of the optimal off-line solution. Hybrid approaches combine both throughput and buffer measurements in the adaptation process, as in the PANDA case [57]. The PANDA client takes inspiration from the TCP congestion control. A target data rate is set by the client to probe the network; the requested bitrate and the time interval between consequent requests are computed to match this target, by also keeping into account the buffer filling level to reduce freezes. The hybrid SQUAD algorithm selects the next quality by primarily minimizing quality switches [102]. A feasible set of possible qualities is created at each decision step by considering only qualities whose expected download time is lower than the segment duration. This constraint is relaxed when the buffer level is above a given threshold.

We refer to the works by Seufert et al., Kua et al. and Sani et al. for an exhaustive discussion on HAS rate adaptation heuristics [50, 84, 87]. As pointed out in Section 1, several QoE factors (as freezes, live latency or fairness) cannot be completely optimized by classical client-based heuristics, which therefore need to be supported by optimizations taking place in the network, at the application or at the transport level. Consequently, this survey mainly analyzes works falling in these three categories, as reported in Section 3.

2.2 QoE Factors in HAS

The ultimate goal of any streaming solution is to provide a good QoE to the end users of the service. Even though a general QoE model for HAS has not been developed yet, it is still possible to identify crucial factors having a strong influence on the QoE of adaptive streaming users, which are discussed in the remainder of this section. The same QoE factors are also used to discuss and compare the different QoE-centric solutions presented in Sections 3 and 4.

Video freezes The first and foremost objective is to avoid video freezes, which have the strongest impact on user experience. Even though the HAS principle was designed to avoid freezes, current implementations are still affected by this problem. For instance, the 2015 Conviva report shows that almost 25% of the analyzed HAS sessions are affected by at least one freeze [21].

Video quality Maximizing the video quality, given a certain available bandwidth, is beneficial for the user experience. Nevertheless, the same Conviva report highlights that low video quality

still impacts 54% of the sessions. A clear trade-off can be identified between reducing freezes and maximizing quality, as requesting high-quality segments increases the chance of rebuffering events.
Quality switches A third objective to take into consideration is the amount of quality switches, as the adaptation heuristic can dynamically change the video quality to accommodate bandwidth variations. The impact of quality switches on user experience has been investigated by Hossfeld et al. [40]. The authors show that the time spent on the highest quality has the strongest impact on user experience, more than the number of switches itself. This result has also been confirmed in recent work by Tavakoli et al. [94]. This study also highlights that sudden reductions of video quality are perceived negatively by the user.
Latency In recent years, HAS has gained consistent momentum for the streaming of live events. In this scenario, end-to-end latency is of extreme importance to avoid the so-called *spoiler effect*. In fact, regular cable or satellite streams are usually characterized by a 5 to 10 seconds latency, which can instead grow to 30–60 seconds for HAS. This aspect entails that the viewing experience of HAS users can be spoiled by cable and satellite users, especially in case of important events. HAS solutions have been originally developed for Video-on-Demand (VoD), where latency is less of an issue, and are therefore suboptimal when it comes to live streaming. Together with live latency, also startup latency should be reduced as much as possible, especially when a user quickly switches between different video channels to search for some interesting content to watch.
Fairness Akshabi et al. are the first to report suboptimal behavior of HAS clients competing for shared bandwidth [5]. Fairness issues are not due to TCP dynamics, but mainly arise from the rate adaptation algorithms, as they decide on the actual rate to download. When multiple clients stream video at the same time, wrong bandwidth estimations can occur, due to the temporal overlap of the activity-inactivity periods of different clients. This wrong estimation subsequently affects the bitrate selection and thus the final QoE. Even though fairness is a system-wide characteristic rather than a user perceived QoE factor, it is often a desired property of the system, especially from the network and service provider point of view. Nevertheless, fairness issues can finally degrade user experience and should therefore be minimized.

Several attempts have been carried out to combine the aforementioned factors in one single combined QoE model for HAS [25, 40, 94]. Some of the proposed QoE models are also used in conjunction with network-based algorithms to improve the delivery of HAS streams [7, 26, 29, 33, 63, 101], which will be discussed in Section 3. De Vriendt et al. propose a model that is a linear combination of the average requested quality and its standard deviation, to keep into account quality switches [25]. The model is based on results obtained via a crowdsource experiment. A similar modelling approach is taken by Bentaleb et al. [7]. Besides the average quality and average quality difference between consecutive segments, the authors also include in their model a linear decreasing term depending on the number of freezes and the initial startup latency. Wang et al. propose an online QoE model, which is a function of the logarithm of the segment bitrate and the inverse of the freeze time [101]. The QoE model proposed by Mansy et al. depends instead on the screen resolution of the device, the viewing distance and the resolution of the quality played by the client [63]. Conceptually, the user-perceived quality degrades when the video resolution is lower than the screen resolution. Moreover, the structural similarity index is included in the model to account for the effect of the encoding bitrate. Essaili et al. employ a simple QoE model that is a linear function of the peak signal-to-noise ratio [29].

As a consensus on a specific QoE model has not been reached yet, we will use the aforementioned individual QoE factors when discussing existing works in Section 3, and to provide general guidelines on the best approach to use in Section 4.

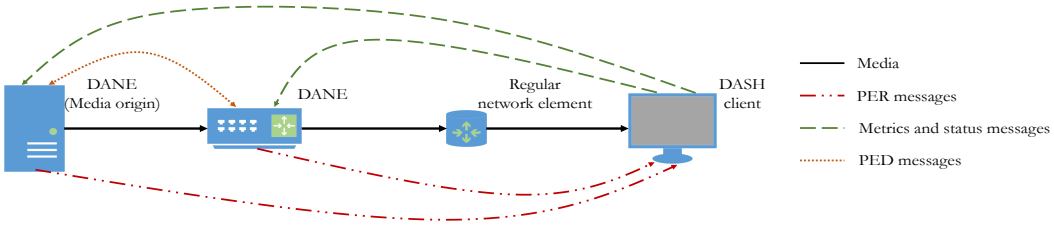


Fig. 3. In an MPEG-SAND architecture, DASH-Aware Network Elements (DANE) can communicate among each other and with the clients to improve the end-to-end delivery of the video (adapted version of [95]).

3 STATUS OF QOE-CENTRIC MANAGEMENT FOR HAS

In this section, we review the main works addressing the optimization of HAS services. We focus on three different research areas that go beyond purely client-based rate adaptation heuristics, namely: (i) server- and network-based solutions (Section 3.1), (ii) application level optimizations (Section 3.2) and (iii) TCP-layer modifications and emerging network architectures (Section 3.3).

3.1 Server- and Network-Based Optimizations

One of the success factors of the HAS principle has been its pull-based and decentralized nature, which allowed for an easy deployment. As reported in Section 2.2 though, this approach might not be sufficient to fully optimize users' QoE. For this reason, several approaches have been proposed, which use additional nodes located inside the network to help the delivery of the video [8, 19]. Following this trend, an extension of the MPEG-DASH standard has recently been published, called Server And Network assisted DASH (SAND) [43, 95]. In an MPEG-SAND architecture (Figure 3), DASH clients can communicate with DASH-Aware Network Elements (DANE). DANE nodes are located between the DASH clients and the server (which might be a DANE node itself), and are aware of the underlying video streaming traffic. Based on the streaming and network conditions, these nodes can take decisions on the best way to handle DASH traffic in order to improve streaming performance (e.g., via bandwidth reservation, prioritization, rerouting, intelligent caching etc.). The MPEG-SAND standard defines the so-called SAND messages that can be exchanged between DANE elements and DASH clients. SAND status messages are used by the clients to report information to the network, so that the DANE elements are aware of the status of the clients. As an example, DASH clients can report QoE metrics that can be used by the network for monitoring purposes and/or to implement QoE-aware optimizations. Status messages also comprise information on the desired quality and bandwidth to facilitate resource sharing among competing clients, and hints on the future segments to request, to enable efficient caching strategies at the DANE nodes. Parameters Enhancing Reception (PER) messages are instead sent from the network to the clients in order to enhance and improve their quality adaptation. A DANE element can inform the clients about the available network throughput or communicate the segments that are already cached by the DANE. A client can therefore prefer to request these segments, as they are already available at the DANE and can be downloaded faster. Finally, DANE elements can communicate among each other using Parameters Enhancing Delivery (PED) messages. A PED message can, for example, be used by the server to communicate information about the streamed video to the network delivery elements. All SAND messages are delivered using the extensible markup language format over HTTP and follow a specific syntax defined by the standard [43].

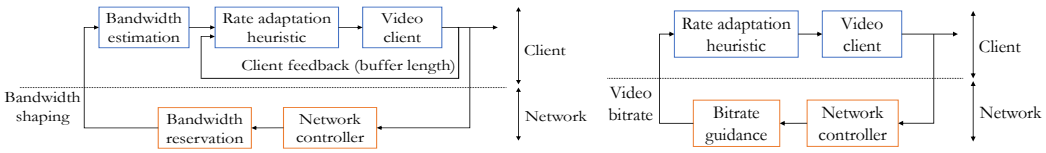
Compared to client- or server-based solutions, network-assisted approaches are more difficult to deploy as they require: (i) a modification of the network nodes and (ii) an active collaboration between service and network providers. The MPEG-SAND standard represents an important enabler for these solutions, as it provides the set of messages the network, servers and clients can interchange to optimize the delivery of the video. Moreover, this collaboration can be beneficial for

all actors involved, in terms of generated revenues and reduced user churn, as pointed out from a techno-economic perspective by Ahmad et al. [3].

Being a fairly recent development of the MPEG-DASH standard, not all the works reported in this section follow the MPEG-SAND architecture. Nevertheless, they all share the same general principles and objectives defined by the standard, where a set of network nodes can communicate with each other and with the clients in order to optimize the end-to-end delivery of the video. In the following, we review the main works in this domain and categorize them based on the main approach used to improve the video delivery.

3.1.1 Traffic rerouting. In approaches exploiting traffic rerouting, the performance of the paths connecting server and clients is continuously monitored. When certain conditions are detected (e.g., increased packet loss or congestion) a path recalculation occurs to reroute the video traffic and guarantee good streaming performance. Egilmez et al. have been among the first to investigate the use of rerouting for layered adaptive streaming [28]. A network controller, implemented using the Software-Defined Networking (SDN) principle, monitors the packet loss and delay of the network links. Each second, the optimal path for the video clients is computed, which minimizes a cost function based on packet loss and end-to-end delay. Two different optimization strategies are adopted for layered streaming. In the first solution, only the base layer is rerouted, as it is required at client-side to decode the video, while the enhancement layers are delivered over the shortest path. This strategy is sufficient to achieve higher quality compared to a shortest path delivery, when congestion is low. In the second solution, the enhancement layers are rerouted as well (with a lower priority compared to the base layer), which is effective when network congestion is high. Different service classes can also be introduced in the path computation optimization problem for layered video streaming [6]. In this work, network switches are equipped with one queue for each class, and the goal of the network controller is to maximize the service provider revenue, by rerouting traffic to satisfy as many high service class users as possible. While these rerouting strategies are executed periodically and for all the video flows in the network at the same time, Cetinkaya et al. recompute the optimal path per client, each time a new segment request is issued [13]. The server communicates to the network controller the bitrate of the requested segment. Based on this information and the links status, the controller reroutes the traffic to maximize the overall links throughput. This strategy is able to almost completely eliminate freezes compared to a best-effort delivery and to reach 15% higher bitrate. Despite that, scalability issues can emerge, as the controller has to operate on a per-client basis. To solve this problem, the path recomputation can also be triggered by the client itself [68]. In this case, when a client experiences a rebuffering event, an SDN controller collects measurements on network links (in terms of bandwidth, packet loss and jitter) and runs a shortest path selection algorithm to find the best routing path for each flow. This approach can reduce the number of switches compared to a best-effort delivery and increase the time spent on the highest quality by 10%.

Traffic rerouting can also be indirectly achieved when multiple servers are available. In this scenario, the client can autonomously and dynamically select the best server to stream the video from [10, 101]. Bouten et al. use network characteristics to perform this selection [10]. During the start-up phase, the client polls the capabilities of the different servers by downloading a segment from each of these servers. The client then starts streaming from the server providing the best throughput. To avoid ending up in a local optimum, a probabilistic search is carried out by the client to test the performance of the available servers. This search is only executed when the buffer filling level is above a given threshold, to reduce the risk of freezes. The server selection strategy proposed by the authors is 12% worse than the optimal one in terms of QoE, which is modeled as proposed by De Vriendt et al. [25]. The user's QoE can be taken directly into account



(a) Bandwidth shaping (b) Bitrate guidance

Fig. 4. When bandwidth shaping is used (a), the network enforces a specific bandwidth for each client. In the bitrate guidance scenario (b), the network provides an explicit bitrate to the clients (adapted version of [19]).

in the selection process [101]. The authors present an online QoE model, based on the bitrate of the segment and the freeze duration, which is used to evaluate the available servers. During the streaming session, the client evaluates the servers in terms of achieved QoE, and selects the one providing the best performance. Clients can also share QoE measurements among each other to enhance their selection process. Compared to a standard DASH client, the proposed approach results in 20% higher QoE.

Highlights for traffic rerouting: Rerouting allows to dynamically select the best path or server for the video delivery. As such, the video is always streamed over high-throughput, low-latency links, which increases the achieved video quality and reduces the amount of freezes. Relevant references: [28], [6], [13], [68], [10], [101].

3.1.2 Bandwidth shaping and bitrate guidance. Bandwidth shaping and bitrate guidance techniques have attracted a lot of attention in the adaptive streaming community [19, 36]. Most of the works in this area share a similar architecture (Figure 4). A centralized node can collect information from both the network nodes and the clients. Network measurements include the available bandwidth and the number of streaming clients [7, 19, 36, 46, 63, 65, 73]. Client-based information includes, for example, the screen dimension [63], the device type [36], the bitrates of the video [46] and buffer filling level [7]. The centralized node has therefore a comprehensive view of the streaming service, and can select the best bitrate for each client in order to maximize an objective function, which often models the users' QoE. The SDN principle is often used to provide a practical and scalable implementation of the developed solutions. Cofano et al. formally compare the performance of bandwidth shaping and bitrate guidance techniques [19]. A centralized node computes the optimal bitrates for the clients in order to obtain fairness in terms of video quality, estimated using the SSIM index. When bandwidth shaping is used, clients with similar optimal bitrates are assigned to a bandwidth slice. Despite this, the clients keeps the sole responsibility for the quality adaptation. In the bitrate guidance case, the optimal bitrates are explicitly communicated to the clients that download the corresponding segment. This approach provides the best overall results in terms of fairness and switching frequency compared to the bandwidth shaping approach. Similar conclusions are also drawn by Kleinrouweler et al. [46], who show that bitrate guidance is sufficient when streaming traffic is predominant. When DASH players have to compete with cross-traffic applications, the combination of slicing and guidance can provide the best results. Complex QoE models can be employed to improve the centralized bitrate selection [7, 63]. Bentaleb et al. use a QoE model depending on the average video quality, number of switches, freezes and startup latency [7]. The network controller selects the best video bitrate to maximize the QoE for each client. The optimal values are sent to the clients, which use it as an upper bound in their quality adaptation. The QoE model proposed by Mansy et al. depends instead on the screen resolution of the device, the viewing distance and the resolution of the quality played by the client [63]. Conceptually, the user-perceived quality degrades when the video resolution is lower than the screen resolution. Moreover, the SSIM is included in the model to account for the effect of the

encoding bitrate. A measurement proxy located between the server and the client is investigated by Mok et al. [65]. The proxy estimates the highest available bitrate the clients can download based on the network conditions. This measurement is carried out using the media traffic itself, i.e., no explicit probing is needed. The clients use the quality level provided by the proxy as an upper bound. Moreover, a new adaptation heuristic is proposed to limit the number of switches. Similarly, Petrangeli et al. design a system of coordinated network proxies to help HAS clients achieving fairness in a multiple-bottleneck network scenario [73]. The proxies compute an estimate of the fair bandwidth share of each client streaming video, and communicate this information to each client. The clients then request a quality that can both optimize their own QoE and the fairness of the whole system.

Highlights for bandwidth shaping and bitrate guidance: A centralized controller computes the optimal bitrates the clients should request, based on information collected from both the network nodes and the video clients. This decision is enforced by either creating a network slice or by informing the clients on the quality to request. These approaches allow to explicitly control the video quality the clients can achieve and improve the fairness of the system. Relevant references: [19], [36], [46], [7], [63], [65], [73].

3.1.3 Cross-layer optimization for mobile networks. When the last mile of the mobile network can be controlled, several optimizations can be carried out, by exploiting video-aware packet scheduler algorithms at the radio interface [14], channel and power allocation problems [109] or by monitoring the radio conditions [29]. In this case, the specific conditions and constraints of the mobile environment have to be taken into consideration when managing the radio resources. Chen et al. propose the AVIS system, which is composed of two elements [14]. An allocator centrally decides the bitrates to be requested by the clients, by maximizing an aggregate utility across all users while respecting the total number of resource transmitting blocks available at the base station. The utility is simply modeled as a logarithmic function of the requested bitrate and the number of quality switches. An enforcer schedules the transmission of the video flow packets so that the rate decided by the allocator is respected. When the AVIS systems is used, fairness among competing clients can be improved. Improving fairness is also one of the main objectives of Zhao et al. [109]. The authors define an optimization problem to allocate each user to a specific channel and transmission power level, in a layered video streaming scenario. More wireless resources are allocated to users whose segment playback deadline is close to expire or when the video quality contribution of requesting a specific layer is high. In other words, the marginal video quality contribution of higher layers decreases as the number of layers increases. Moreover, a DASH proxy stabilizes the quality decision process of the clients. The proxy computes the optimal quality level to request in order to maximize channel utilization and reduce switches. If the quality requested by the client is higher than the optimal one, the proxy overrides the client's decision by rewriting the HTTP GET request. A different approach for layered streaming is used by Fu et al. and Deng et al. [26, 33]. Video layers are assigned a priority based on the content characteristics. This priority depends on the visual quality gain of the layer and the bandwidth cost of transmitting the layer itself. The priority marking can happen at the server or in the network. The wireless base station can drop lower priority layers based on the channel condition of each user. This approach does not require modifying the radio interface scheduler and can provide 20% QoE improvements, with QoE modeled as a function of the number of received layers at the client. Essaili et al. use a QoE optimizer, located in the LTE network, which decides the transmission rate for each streaming client in order to maximize QoE, which is a function of the rate itself [29]. A penalty term is introduced to stabilize the rate and avoid QoE oscillations. Once the rate is decided, a proxy rewrites the HTTP GET request of the client to match the optimal rate. The easiest strategy is to choose the

closest video representation for which the bitrate is below the optimal rate. The authors propose more complex solutions that take into account the buffer filling level of the clients. Compared to a classical HAS system, this approach results in up to 35% better QoE for the worst-case user.

Highlights for cross-layer optimization: Cross-layer optimizations take the condition of the radio interface into account to improve the delivery of HAS streams. Many of these solutions employ ad-hoc packet scheduling algorithms to improve the video quality delivered to the clients. As the optimization is carried out by a centralized node that is aware of the clients' conditions, fairness is also improved. Relevant references: [14], [109], [29], [33], [26].

3.1.4 Stream prioritization. Prioritization has proven to be an effective way of optimizing QoE, especially in terms of video freezes [75]. Pu et al. propose a proxy node for mobile networks to concurrently optimize the delivery of multiple DASH streams [76]. DASH streams are assigned a different priority depending on the requested quality, which is considered a good indicator of the client status. Particularly, when the requested bitrate is below a specific threshold, which might indicate that the client is close to a freeze, the wireless scheduler assigns the stream a higher priority. The buffer status of the client can also be used to trigger prioritization [110]. In this work, a network controller prioritizes the packets belonging to video flows using a dedicated queue, when the client buffer drops below a certain threshold. The authors compare the performance of the priority queuing approach and a weighted fair queuing approach, and show that the latter performs better as it enables a more accurate bandwidth allocation to the flows. A similar approach is used by Petrangeli et al. [75]. A centralized SDN controller intercepts the HTTP GET requests of DASH clients and estimates their buffer status and requested quality using a machine learning algorithm. Based on this information and network measurements to detect congestion, video segments are enqueued in a prioritized queue to avoid future freezes. This approach can reduce freeze time with 45% when compared to classical HAS heuristics, without severely impacting the performance of cross-traffic applications.

Highlights for stream prioritization: A network controller is in charge of temporarily prioritizing the delivery of HAS streams, based on information as the requested quality or the clients' buffer level. These approaches are extremely effective in reducing video freezes. Relevant references: [75], [76], [110].

3.1.5 Content delivery network orchestration and caching optimization. To satisfy the huge demand of video streaming requests, streaming providers have massively adopted Content Delivery Networks (CDN), where the video content is locally and temporally stored. By bringing the content closer to the end users, it is possible to reduce the load on the origin server and serve the video with lower latency and increased bandwidth. Casas et al. perform an experimental measurement study on the performance of YouTube's CDN infrastructure, with traffic collected in the core network of a national-wide mobile operator [12]. YouTube CDN servers are located at multiple autonomous systems, with some deployed directly inside the Internet Service Provider (ISP) networks. The measurement study shows that less than 10% of the flows served from CDN nodes inside the ISP network achieve a download throughout lower than 1 Mbps, while this value increases to 90% for servers located outside the ISP network. A large-scale measurement study is instead performed by Boettger et al. to model the CDN infrastructure of Netflix [9], called Open Connect. Particularly, the authors discover that half of Netflix's CDN nodes are deployed within ISPs, while the remaining nodes are deployed withing Internet Exchange Points (IXP). CDN nodes are observed in 569 different ISP and 52 different IXP locations, respectively. Moreover, when looking at the deployment of the Open Connect platform over time, Netflix seems to mainly rely on IXP deployment as a first step, followed by a fine-grained ISP deployment. The authors argue that this strategy allows Netflix to

reach a large user base during initial deployment, using IXP located servers. ISP located servers are instead used in a second phase to effectively reach a larger user base, which is often geographically scattered. A CDN orchestrator can be employed to manage such complex and geographically distributed infrastructures, as investigated by Mukerjee et al. [67]. The authors propose a control orchestration plane to optimize Conviva's C3 architecture, an Internet-scale CDN platform for the delivery of live videos [34]. A centralized controller, which operates on a timescale from tens of seconds to minutes, computes the optimal distribution trees for all the clients and videos in the different CDN clusters. The individual clusters apply a distributed algorithm on a sub-second timescale, to update the video forwarding strategy and keep into account local changes (e.g., link failures, workload changes). General caching strategies for video streaming can take into account the user and content characteristics (e.g., user mobility, content popularity) to better place the content in the CDN edge caches [62]. Krishnappa et al. exploit a particular user behavior to improve the caching efficiency of YouTube videos [48]. Users are more inclined to watch videos on top of the related video list available on the YouTube web page. By rearranging this list to give preference to cached videos, cache hit rate is improved by a factor of 5. Caching strategies can also be improved when future user requests are known in advance [18]. For example, binge-watching has become typical on popular video streaming services, meaning that users tend to watch many episodes of the same TV show consecutively. This information can be used to estimate future video segment requests and improve the content placement in CDNs.

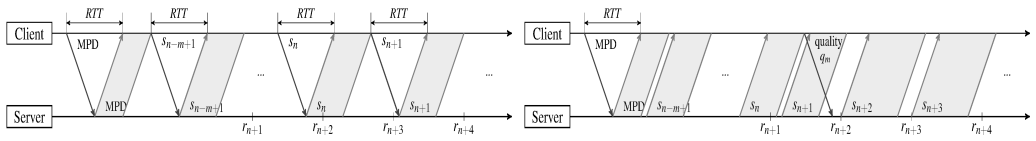
Unfortunately, caching can also negatively interfere with the particular dynamic of adaptive streaming clients, leading to frequent bitrate oscillations [54]. When a cache hit occurs for the requested segment, the throughput perceived by the client is generally high, which brings the client to request the next segment at a higher quality. If a cache miss occurs, the segment will have to be retrieved from the origin server. The increased latency in the segment delivery results in a lower bandwidth, which leads to a downshift in the requested quality. Transcoding can reduce this issue, by generating on-the-fly lower bitrates, starting from segments at higher bitrates located in the cache [1]. Transcoding techniques are examined in detail in Section 3.1.6. The problem of bitrate oscillations and caching is analyzed by Ge et al., in the context of edge computing in a 5G network [35]. A centralized node collects information on the requested segments and the conditions of the radio interface, and determines the most appropriate representations of individual segments to be cached. Only segments whose bitrate does not exceed the downlink capacity experienced by the mobile users are cached. Moreover, the replacement strategy tries to ensure segment continuity if a video is cached, in order to avoid cache misses. This approach can provide similar quality to the end users compared to a least frequently used cache strategy, but with 50% less switches. Li et al. analyze the problem of content placement in a wireless scenario, where the clients can stream the content from a set of edge servers, located close to the users [55]. The authors assume that the edge servers have to store all the segments belonging to a certain representation, if that particular representation needs to be stored. Each user in the system is associated with a distortion function, which decreases as the bitrate of the requested segment increases. The problem is therefore how to place the video representations in the edge servers in order to minimize the distortion experienced by the clients, assuming videos have different popularities and distortion functions, and the edge servers have limited cache capacity. Distortion is reduced by 20%, compared to a naive caching strategy that only caches most popular videos. A joint prefetching and caching architecture, called iPac, is proposed by Liang et al. [58]. A cache manager generates prefetching requests based on the segment requests from the clients. The bitrate of the prefetched segments is fixed to the one requested by the client. Moreover, the number of prefetched segments is limited to avoid wasting network resources if the requested bitrate would change. A prefetching manager decides whether

the prefetching request should be forwarded to the origin server or not. It is worth noting that not all the prefetching requests can be accepted, as they have to compete with the same bandwidth as standard cache miss requests, which have higher priority. This decision is based on how likely the prefetched segments will be requested in the near future. The authors propose an online prefetching algorithm that can perform at least half as good as an offline algorithm with complete knowledge of the system, in the worst case scenario. The iPac architecture can reach eight times higher byte-hit ratio compared to a least recently used cache strategy and increase user throughput by 50%.

Highlights for CDN orchestration and caching optimization: CDN orchestrator and intelligent content placement algorithms can be used to bring the most adapted video content (e.g., the most popular or the most personalized) closer to the end users. These solutions guarantee a higher throughput for streaming, which results in higher video quality and less freezes. Nevertheless, the succession of cache hit and cache miss complicates the bandwidth estimation process of the clients and can negatively affect the number of quality switches. Relevant references: [12], [9], [67], [62], [48], [18], [54], [1], [35], [55], [58].

3.1.6 Server-assisted delivery. When the control on network components is limited, the server itself can assist the delivery of the video. Akhshabi et al. investigate a server-based traffic shaping algorithm, designed to reduce the quality fluctuations due to competing HAS clients sharing the same bottleneck [4]. The server identifies instable behavior of the clients (e.g., frequent quality increase/decrease) and limits the available download bandwidth to match the rate of the requested segment. This approach can eliminate the on-off pattern of the client requests, which is the main responsible for instability and unfairness in HAS. The server can also communicate directly with the client in order to efficiently schedule the client's HTTP GET requests to avoid rebuffering events [20]. The server foresees the occurrence of freezes by computing the download time of future segments over a specific time window. If a freeze is detected, a signal is sent to the client that consequently tries to ramp-up its buffer by downloading multiple segments in sequence. This approach reduces freezes up to 50% in wireless environments, compared to a client-based solution.

Many other works focus instead on the optimal choice of the encoding bitrates and online transcoding operations [90, 97]. In online transcoding, only few video representations are prepared by the server before the content is published; all the others are prepared on-the-fly when and if the clients request them. This approach allows to reduce the storage requirements in adaptive streaming, at the cost of an increased risk of rebuffering events, as the content has to be prepared online. Moreover, online transcoding is very computational intensive. Song et al. develop a scheduling algorithm for power-efficient transcoding tasks [90]. An optimization problem is formulated to assign each transcoding job to the right CPU, whose frequency is adjusted to save energy while meeting the transcoding deadline. This approach can reduce power consumption up to 31% compared to the CPU scaling frequency algorithm used by Linux servers. To guarantee that the transcoding deadline is met when a client requests a segment, Ma et al. use a video transcoding time (VTT) estimation, based on a measurement study. When a video is uploaded to the server (e.g., in case of user-generated content), the encoding tasks are sorted based on the corresponding VTT and inserted into a low priority queue. Only when a segment is actually requested, it is moved into a high priority queue to guarantee a timely transcoding. Content popularity can also be taken into account [45]. In this work, all quality representations of videos whose popularity is higher than a rank threshold x_{high} are stored in edge servers. The highest quality only is stored for videos with popularity rank between x_{high} and x_{low}, in order to allow online transcoding. All the other videos are only available in the origin server. An optimization problem is formulated to minimize the storage and transcoding costs at the edge servers, and minimize bandwidth utilization between the origin and the edge servers when the content cannot be served by the latter. Using this approach, it



(a) HTTP/1.1 (b) HTTP/2 server push

Fig. 5. In classical HTTP/1.1 (a), each segment has to be retrieved independently and sequentially by the client. In HTTP/2 server push (b), the server can automatically push segments back-to-back. This behavior eliminates lost RTTs between segment requests and increases bandwidth utilization (from [100]).

is possible to reduce operational costs by 50%, when all the content is cached in advanced or is transcoded in the edge servers. Krishnappa et al. use a Markov model to predict which segments are going to be requested in the near future [49]. They also propose a hybrid transcoding policy, where only the first segment of the video is pre-encoded at all quality levels, while the remaining segments are transcoded online. The combination of this strategy and prediction allows to limit the rebuffering ratio to less than 1%. Moreover, transcoding operations are reduced by a factor of 10 compared to a solution where all the segments are pre-encoded. Joint transcoding and delivery strategies have also been proposed in [103]. A centralized node collects clients' preferences in terms of servers downstream bandwidth and estimates the number of requests for a particular segment, based on the segments requested in the near past. Based on this information, clients are redirected to more performing servers, while segments that are more likely to be requested are prioritized in the transcoding process. In a simulated environment, almost 45% of the clients are able to stream the video at a higher bitrate compared to a solution where the complete video is pre-encoded.

Highlights for server-assisted delivery: Server assistance is used to reduce client instability and therefore reduce the number of quality switches. Transcoding operations allow to reduce the storage requirements in CDN nodes, as the content is prepared on-the-fly if requested, but increase the risk of video freezes. Relevant references: [4], [20], [97], [90], [45], [49], [103].

3.2 Application Level Optimizations

In case of over-the-top delivery, network elements are not accessible and cannot be used to support video clients. In this section, we review existing works that modify the application level of the HAS clients to improve the video delivery. We refer to the application level in a broad sense, both considering modifications of the actual application layer (e.g., HTTP/2 instead of HTTP/1.1) and enhancements of the client rate adaptation heuristic. Nevertheless, all the works in this section share two common aspects: (i) the main optimization is located at the client and (ii) they are compatible with any rate adaptation heuristic. In other words, these solutions can run on top of any HAS client implementation.

3.2.1 Adaptive streaming over HTTP/2. The HTTP/2 protocol was originally developed to reduce the loading time of web pages, but it has also been applied to the delivery of multimedia content. Among the main features of HTTP/2, it is worth citing the native support of multiplexing (as opposed to HTTP/1.1), the possibility for the client to terminate the download of specific content and for the server to push content to the client, without an explicit request from the latter. Mueller et al. have been the first to analyze the impact of HTTP/2 in adaptive streaming [66]. Simply replacing HTTP/1.1 with HTTP/2 does not bring major improvements to the streaming performance and bandwidth utilization. Moreover, using HTTP/2 causes a slightly increased overhead compared to HTTP/1.1, as HTTP/2 is encrypted by default. By exploiting the specific functionalities of the HTTP/2 protocol, it is instead possible to consistently improve streaming performance. Huysegems et al. review the possible strategies that can be adopted in the context of adaptive streaming over

HTTP/2 [42]. They propose ten different strategies based on stream termination, multiplexing and server push. Particularly, in server push, the server can proactively send (push) segments to a client based on previous requests (Figure 5). Especially for live streaming, using a push-based approach has multiple advantages. First, because subsequent segments can be pushed back-to-back, lost RTT cycles between such segments are avoided, thus increasing the average link utilization and video quality. Second, a reduction of the live latency can be obtained, because video data is pushed as soon as it becomes available at the server. This is in contrast with traditional pull-based HAS approaches, where the client has to make an explicit request for new segments. In light of these advantages, the server push performance has been extensively investigated in order to reduce the live latency and the initial delay in HAS [15, 69, 100, 104, 105, 107]. Wei et al. have been the first to investigate the performance of HTTP/2 server push in the context of adaptive streaming [104]. By reducing the segment duration from 5 to 1 second, the authors are able to reduce live latency by a factor 3. Despite this advantage, a trade-off is present between the number of pushed segments and streaming performance. Pushing many consecutive segments improves bandwidth utilization but reduces the capability of the clients to adapt to varying network conditions. To solve this issue, van der Hooft et al. propose to limit the number of pushed segments in flight to a certain value K [100]. This value is set based on the segment duration of the video and the network RTT. By combining this approach with segments with a sub-second duration, the startup and live delay are reduced by 30% compared to HTTP/1.1, with minimal impact on the freeze time. A similar approach is also proposed by Xiao et al. [107]. This study exploits a previous finding showing that pushing many consecutive segments can reduce the energy consumption up to 18% compared to a pull-based scenario [105]. Consequently, K is set to minimize the energy consumption on mobile devices, while the bitrate of the pushed segments is chosen to guarantee that the buffer does not drop below a given threshold. Moreover, stream termination is used to stop the push cycle in case a bandwidth drop is detected, and start a new one at a lower quality. This approach allows to eliminate video freezes and obtain 15% higher quality compared to classical adaptive streaming over HTTP/1.1. The buffer filling level can be included directly into the computation of the K value [69]. In this work, the optimal K is set in order to keep the buffer filling level above a specific threshold. To improve the bandwidth estimation during a push cycle, Cherif et al. use a WebSocket channel between server and clients, to communicate the beginning and the end of a push cycle. This approach is extremely beneficial when the content is retrieved from the local cache (as in a browser implementation), which can cause an overestimation of the available bandwidth.

Highlights for adaptive streaming over HTTP/2: The most exploited functionality of the HTTP/2 protocol is the server push mechanism, which results in higher link utilization compared to HTTP/1.1. This aspect results in a higher video quality and reduced live latency when server push is combined with sub-second segments. Nevertheless, these approaches might increase the number of freezes, as pushing segments reduces the bandwidth adaptation capability of the clients. Relevant references: [66], [42], [100], [105], [107], [69], [15].

3.2.2 Meta-heuristics for increased client-awareness. Streaming performance can be enhanced when the client can exploit additional information in the rate adaptation. This information, often referred to as context, can include positioning data (e.g., GPS) or historical information on the available bandwidth [59, 82, 98]. As an example, a mobile streaming client can be made aware of regions with limited or zero-bandwidth conditions, as a tunnel, to proactively avoid video freezes. In the event of future scarce connectivity, the client can ramp-up its buffer in order to compensate this condition. Liotou et al. propose an algorithm that deliberately degrades the quality requested by the client to ramp-up its buffer and provide a continuous playout in the low-connectivity zone [59]. When historical data on the available bandwidth are coupled with GPS position information, the

bandwidth prediction of adaptive streaming clients is greatly improved, as shown by Riiser et al. [82]. The GPS and bandwidth information can be used to foresee future connectivity drops along a specific route, thus helping the actual quality adaptation. Particularly, the client uses both past perceived bandwidth samples and historic data to better plan the next quality to download. This approach results in a more stable quality behavior and fewer freezes. Optimizing the parameters of a given heuristic on-the-fly can also be beneficial [98]. In this work, a learning agent is designed to learn the best configuration of the heuristic parameters based on the bandwidth characteristics. As an example, the heuristic can be made more aggressive when bandwidth conditions are good and stable, or more conservative otherwise.

Highlights for meta-heuristics: Meta-heuristics exploit context information (as GPS and historical data on the available bandwidth) to improve the bitrate selection of the client. These works try to reduce the occurrence of freezes by using the context information to anticipate future bandwidth drops. Relevant references: [59], [82], [98].

3.2.3 Client-based prefetching. The performance of prefetching algorithms has been investigated in the scope of multi-view video streaming, where a user can dynamically switch between different video channels (e.g., different channels or different cameras capturing the same event). The goal in this scenarios is to minimize the channel switching delay [11, 47]. Prefetching strategies should carefully balance two contrasting objectives, namely maximizing quality and avoiding stalls in the played stream and prefetch new content from different channels to provide a seamless switch. Krishnamoorthi et al. exploit the typical on-off pattern of adaptive streaming clients for their prefetching algorithm [47]. Particularly, prefetching only occurs during off periods, to avoid interfering with the streaming of the played content. In the simplest solution, prefetching only occurs when the buffer filling level is above a certain minimal threshold. This approach can be improved if the moment a certain channel is going to be played by the user is known in advance (e.g., by using a recommender system). In this case, prefetching can occur when the player is in an on period as well, to guarantee that the first segments of the new channel are already downloaded when the user would switch. This deadline-based strategy comes at the cost of a lower playback quality for the played video content and more freezes when bandwidth is scarce. To avoid this behavior, Carlsson et al. explicitly split the bandwidth between the played stream and the alternative streams [11]. Particularly, the currently played stream is allocated more bandwidth based on the buffer filling level, to avoid freezes. Given the bandwidth budget for prefetching, an optimization problem is proposed that maximizes the quality of the prefetched segments, assuming channels have different switching probabilities. Once the quality of the played and alternative streams is selected, the segment download is scheduled in a round-robin fashion. Conversely, prefetching can also be used in classical streaming scenarios to increase the video quality. This behavior has been experimentally observed by Sieber et al. for the browser-based YouTube player [88]. If a low quality level is requested but the adaptation heuristics detects an increase in the available bandwidth, one or more higher quality levels are prefetched in parallel, to present the user with a higher quality as soon as possible. Unfortunately, this approach results in almost 33% redundant traffic in the worst-case scenario.

Highlights for client-based prefetching: Prefetching algorithms download in the background video content that is likely going to be watched in the near future. This behavior minimizes latency in the context of channel-switching. Video quality and freezes can be negatively affected though, as prefetched content is in competion with the played stream. Relevant references: [47], [11], [88].

3.3 Transport Level Optimizations and Emerging Network Architectures

The TCP protocol was not originally designed to deliver video streaming applications, which are now responsible for a large portion of Internet traffic. As an example, Esteban et al. [30] show that HAS clients are often not capable of fully utilizing the available bandwidth of a link and that the on-off pattern of adaptive streaming can have a negative influence on the TCP congestion control. In this section, we review studies that optimize the transport layer to improve streaming performance. Moreover, we also analyze works that propose and exploit new disruptive network infrastructures, as the Information Centric Networking (ICN) principle.

3.3.1 TCP solutions for HAS. Enhanced TCP solutions for adaptive streaming generally follow two main trends. First, additional application-level information can be shared with the transport layer to improve the scheduling of TCP packets, in order to primarily avoid freezes [17, 61, 64]. Second, Multi-path TCP (MPTCP) is employed to improve the aggregate throughput of the streaming client [22, 37, 44]. Lu et al. propose to improve the congestion detection mechanism of TCP [61]. In its standard implementation, TCP interprets all packet losses as an indication of network congestion. In wireless networks though, this behavior is sub-optimal, as losses can be due to wireless transmission errors rather than actual congestion. For this reason, the authors propose an algorithm that dynamically decides whether the TCP congestion control should be triggered or not. The algorithm exploits the inter-arrival time of TCP packets, which exhibits different characteristics based on the actual cause of the packet loss, namely congestion or wireless errors. Moreover, normal congestion control is triggered only if the lost packets contain an important frame of the video (i.e., I-frames or P-frames). The proposed approach results in 20% higher throughput than standard TCP. Incorporating the playout deadline of a video streaming packet can also improve the overall performance [17]. In this study, the playout deadline information is used to modify the congestion control parameters of TCP New Reno, in terms of increase and decrease of the congestion window. A flow characterized by an urgent deadline can temporarily adopt a more aggressive behavior, in order to achieve a higher throughput. Using this modification, the video clients can all meet their deadline, by using 15% less bandwidth than with normal TCP. The TCP modification proposed by the authors only takes place at the sender-side, which can facilitate its actual deployment. McQuistin et al. develop a new version of TCP, called Hollywood, that is wire-compatible with standard TCP in order to guarantee deployability [64]. Partial reliability is the main characteristic of this protocol: at the sender-side, TCP packets are retransmitted only if they can actually be used in the playout, i.e., if their playout deadline has not elapsed. If the deadline cannot be met, the payload of the retransmitted packets is substituted by another payload that can be used by the client. At the received side, TCP Hollywood removes head-of-line blocking, so that video segments can be passed to the application layer as soon as they are completed.

When multiple network interfaces are available, MPTCP can be used to increase the aggregate streaming throughput. Despite the logical expected advantages of this approach, MPTCP is not always beneficial for adaptive streaming [44]. The amount of bandwidth on the different interfaces and its variability can in fact have unexpected impacts on the streaming performance. James et al. show that only when the aggregate bandwidth is at least 40% higher than a specific bitrate, an increase in quality can be obtained. Moreover, bandwidth variability on one interface can negatively influence the performance of the whole system. A cross-layer scheduler for multi-path TCP transmissions can reduce these disadvantages [22]. In this study, the cross-layer scheduler, which runs on top of the standard MPTCP packet scheduler, is aware of the structure and relationship among the different frames of the video and knows the buffer status of the client. Therefore, it can estimate the moment when a certain frame would need to be displayed. Moreover, the scheduler knows the conditions of the different interfaces, in terms of RTT, bandwidth and packet loss. More

important frames can then be sent over the interface providing the best performance. By prioritizing the frames that are most likely to be received in time, the proposed solution can improve the average SSIM by 15% compared to the standard MPTCP scheduler. Work presented in [37] shows that user preferences should be taken into account when selecting the appropriate network interface for transmission. In this case, a cost function is associated with each interface, which depends on the specified preference. An optimization problem is defined, which minimizes the total delivery cost while respecting the playout deadlines of the video segments. For example, the cost function can depend on the energy consumption of a specific interface. When a Wi-Fi and an LTE interface are available, a higher cost can be associated to the latter. The authors show that their framework can reduce the amount of data transmitted over LTE by almost 40%, compared to standard MPTCP, without impacting the streaming performance. This result also has a positive impact on the energy consumption, which is reduced by 8%.

Highlights for TCP/MPTCP for HAS: TCP optimizations modify the congestion control and packet scheduling algorithms by incorporating video segments playout deadline information. This approach increases the achieved throughput and, therefore, the played video quality. A similar goal is obtained by MPTCP, which uses multiple network interfaces to stream the video. MPTCP solutions should be carefully designed, as the mutual influence of the different interfaces can negatively impact the amount of quality switches. Relevant references: [61], [17], [64], [44], [22], [37].

3.3.2 ICN approaches. Historically, the Internet has evolved in an ad-hoc manner where incremental patches were added to handle new requirements as they arose. This means that the underlying network model has not changed over the last decades, while the services using the Internet did so drastically. ICN is a disruptive network architecture that moves the traditional focus of a host-oriented communication model to a content-centric model, which can be extremely beneficial in adaptive streaming [52]. Particularly, ICN relies on location-independent naming schemes, in-network pervasive caching and content-based routing to allow an efficient distribution of content over the network. Moreover, ICN nodes can seamlessly use all the available network interfaces to retrieve content, similarly to MPTCP. Content Centric Networking (CCN) and Named Data Networking (NDN) are typical instantiations of the ICN paradigm [2]. Nevertheless, ICN can also complicate the rate adaptation of HAS clients, as pointed out by Lederer et al. [52]. In ICN, the client is not aware of the node serving the content (e.g., the original server or one of the caches disseminated over the ICN network). This aspect complicates the estimation of the available bandwidth, similarly to when caches are used (Section 3.1.5). This issue has been experimentally confirmed by Liu et al. [60]. The authors also show that adaptive streaming over CCN results in 15% higher network overhead than regular DASH. Rainer et al. thoroughly investigate the interplay between different adaptation heuristics and interest forwarding strategies in NDN [78], where an interest represents the client request for a specific content. In this study, a theoretical framework is developed to find the upper bound for the average bitrate the clients can obtain, assuming the network and streaming characteristics are known a priori. Clients streaming over NDN can reach three times higher throughput than in classical TCP/IP networks, independently of the adopted heuristic or interest forwarding strategy. The best performance is reached when multiple interfaces are used to forward an interest and buffer-based heuristics are used at the client. These heuristics are in fact not susceptible to bandwidth miscalculations, which are likely in ICN due to multi-path transmissions and caching. To mitigate this issue, Ramakrishnan et al. investigate the possibilities of network coding in the context of multi-path interest forwarding [80]. In classical ICN, an interest forwarded over multiple interfaces could receive the same duplicated content over each interface. The authors exploit network coding to effectively aggregate the throughput available on multiple interfaces. Further improvements can be obtained when considering that each node has caching

Table 1. Overview of the different QoE-centric strategies. For each QoE factor, it is reported whether a particular approach has a positive (+), very positive (++) or negative (-) impact, or no impact (blank space). The deployment complexity of the solution is also reported (E: easy, M: medium: H: hard).

		Quality	*Switches*	*Freezes*	*Fairness*	*Latency*	*Complexity*	*References*
Sec 3.1	Rerouting	++		+			M/H	[28], [6], [13], [68], [10], [101]
	Bandwidth shaping	++	+/-		++		M	[19], [36], [46], [7], [63], [65], [73]
	Cross-layer optimization	++			++		M/H	[14], [109], [29], [33], [26]
	Prioritization			++			M	[75], [76], [110]
	CDN and caching	++	-	+			M	[12], [9], [67], [62], [48], [18], [54], [1], [35], [55], [58]
	Server-assistance		+	(-)			E/M	[4], [20], [97], [90], [45], [49], [103]
Sec 3.2	HTTP/2	+		-		++	E/M	[66], [42], [100], [105], [107], [69], [15]
	Meta-heuristics			+			E	[59], [82], [98]
	Prefetching	-		-		++	E	[47], [11], [88]
Sec 3.3	TCP/MPTCP	++	(-)	+			H	[61], [17], [64], [44], [22], [37]
	ICN	++	-	++			H	[52], [60], [78], [80],[38], [56], [108]

functionalities in ICN [38]. In this work, a video client can opportunistically retrieve video segments from both the server, using 3G/4G, and from other clients in a peer-to-peer fashion, using Wi-Fi. This solution results in better quality and reduced load on the mobile network. Intelligent caching strategies can also be developed in ICN [56, 108], with similar objectives as in regular DASH. Nevertheless, caching in ICN is advantaged by the naming structure of the interests. This aspect makes it easier for the network to understand what a user is watching and, most importantly, the relationship between different video segments [108]. Yu et al. use network condition information, available at the ICN nodes, and an estimate of future segment requests to prefetch content during off-peak congestion periods. Using this video- and network-aware prefetching strategy, the delivered quality increases by 20%, compared to a DASH system without prefetching.

Highlights for ICN approaches: ICN approaches combine the benefits of a pervasive fine-grained caching infrastructure and multi-path transmissions. These advantages entail a higher throughput compared to standard TCP/IP and, consequently, higher video quality and less freezes. As for caching though, ICN approaches can have a negative impact on the bandwidth estimation of the clients and, therefore, on the amount of quality switches. Relevant references: [52], [60], [78], [80], [38], [56], [108].

4 RECOMMENDATIONS

The goal of this section is to briefly summarize and discuss the different approaches presented in Section 3. Particularly, we provide some general guidelines regarding the benefits (or drawbacks) of each approach on several QoE factors, and the corresponding deployment complexity. A quick outlook of this analysis can be found in Table 1. For each QoE factor of interest, we indicate whether the analyzed approach has a positive or negative influence and the corresponding deployment complexity. It is worth noting that this classification only captures the main common trends among the works in the research areas presented in the previous section. In reality, each work is unique and presents advantages and disadvantages that cannot be completely generalized.

Traffic rerouting can consistently increase the obtained video quality and reduce freezes. To reroute traffic in the network, a centralized element has to know the status of all the network links involved in the delivery of the video, which allows to optimize the end-to-end conditions of the streaming path. This aspect represents a disadvantage of this solution as well, as it is often difficult to have such a global view on the network. Bandwidth shaping/bitrate guidance and cross-layer optimizations guidance techniques represent the ideal solutions when the quality delivered to each user has to be controlled in a fine-grained manner (e.g., to serve premium users or to guarantee fairness). These approaches are usually deployed on a bottleneck link located in the access or

edge network, and are designed to select and enforce a specific quality for the streaming clients. This selection aims to optimize the revenue of the network operator, by maximizing the QoE of particular users, or provide fairness among competing clients. Bandwidth shaping techniques should be carefully designed as they could have a negative influence on quality switches [19, 46]. Cross-layer solutions can be challenging to deploy, as they require modifications of the radio interface. The applicability of bandwidth shaping techniques can instead benefit from the SDN principle and the MPEG-SAND standard. A similar consideration can be repeated for prioritization approaches, which are tailored on the specific task of reducing freezes. By bringing content closer to the users, smart caching strategies succeed in increasing the achieved throughput and consequently increase quality and avoid freezes. Nevertheless, the interference between caching and HAS can have a negative impact on the switching behavior of the clients. This aspect should be carefully considered when designing caching strategies for HAS. Server-based solutions can effectively reduce switches when shaping techniques are used or help reducing storage costs when transcoding operations are adopted. When transcoding is used, the risk of freezes increases as some video segments have to be generated online. Server- and network-assisted solutions are generally characterized by a medium to high deployment complexity, as they require modifications of network nodes. This aspect entails that the streaming provider and the network provider are willing to collaborate to optimize the behavior of HAS clients. Despite this, the presence of the MPEG-SAND standard and the benefits of such a collaboration for all actors involved [3] could be the drivers for an actual deployment of these solutions. Moreover, the MPEG consortium has started exploration activities related to network-distributed video coding and network-based media processing, which will rely on the MPEG-SAND specification [72].

Application level solutions are easier to deploy, as only the client has to be modified. In the case of HTTP/2, also the server has to be updated. Nevertheless, this aspect only marginally complicates an actual deployment, because servers and CDNs are starting providing HTTP/2 functionalities by default. Particularly, HTTP/2 push-based solutions can consistently reduce the live latency, increase bandwidth utilization and video quality, at the cost of reduced bandwidth adaptability that can result in more freezes. Prefetching is beneficial when the channel switching latency has to be minimized. A balance has to be found between prefetching many alternative channels and increasing the quality of the watched stream. Current works in the meta-heuristics domain mostly focus on reducing freezes by exploiting spatial and GPS information on network coverage.

Modifications at the transport layer focus on increasing the throughput achievable by streaming clients, by modifying the congestion control of standard TCP. These approaches also tend to be deadline-aware, i.e., they know when a particular video segment needs to be played at the client, to actively avoid a freeze. On a high level of abstraction, ICN-based solutions are characterized by pervasive in-network caching and multi-link usage. They therefore combine the advantages (and disadvantages) of caching solutions and MPTCP approaches. The optimizations presented in Section 3.3 are the most difficult to deploy nowadays, as they require an upgrade or modification of the current Internet infrastructure. Nevertheless, these optimizations provide important insights for new transport protocols that are currently under development, as explained in the next section.

5 FUTURE DIRECTIONS AND CHALLENGES

As reported in the previous sections, several consistent improvements have been developed to optimize the QoE of adaptive streaming services. Nevertheless, several new challenges are emerging, which will need to be addressed by the multimedia management community in the future. In this section, we identify the main challenges and the newly arising opportunities associated with them. This analysis can help focusing future research in the adaptive streaming domain.

Fig. 6. In tiled VR streaming, the 360° video is divided into spatial regions. Only tiles belonging to the viewport are streamed at the highest quality, to save bandwidth.

5.1 Immersive Video Streaming

Virtual Reality (VR) devices are quickly becoming accessible to a large public. It is therefore expected that the demand for 360° immersive videos will grow consistently in the next years. In VR streaming, the user is immersed in a virtual environment and can dynamically and freely decide the preferred viewing position, called viewport. Unfortunately, VR streaming is often affected by low quality nowadays, due to the high bandwidth requirements of 360° videos. Viewport-dependent solutions have often been proposed for VR streaming, as they are able to reduce the bandwidth required to stream the VR video [77, 83]. In viewport-dependent streaming, only the portion of the video actually watched by the user is streamed at the highest quality. The rest of the video, which is outside the viewport and is therefore less important, can be streamed at a lower quality or not streamed at all. Viewport-dependent streaming can be obtained using online transcoding operations, as foveat-based encoding [83], or by spatially tiling the video [77], as shown in Figure 6. This last possibility is extremely interesting in the HAS domain, as MPEG-DASH has recently standardized a new specification, called Spatial Representation Description (SRD), to support tile-based streaming [70]. In the context of tiled VR streaming, three important questions should be answered. First, *how to prioritize the delivery of viewport tiles as opposed to tiles outside the viewport?* Clearly, tiles in view should be delivered faster than the others. The new HTTP/2 protocol, with its new features as server push, stream prioritization and multiplexing, can represent a possible solution [74]. Second, *how to predict where a user is going to watch in the near future?* When moving, the user can reach regions of the video at lower qualities. It is therefore important to predict these changes in order to provide a seamless transition when the user moves. As an example, Fan et al. propose a recurrent neural network to estimate the fixation point for 360° videos [31]. Third, *what is the effect on QoE of watching regions at different qualities?* Despite the benefits of prediction algorithms, the user can still be presented with a viewport at different quality levels. The impact of this behavior on QoE should be investigated to provide an important input for future tile-based rate adaptation heuristics.

5.2 HTTP Adaptive Streaming over QUIC

As shown in Section 3.3, several advantages can be obtained when the transport layer is modified to better support streaming traffic. Unfortunately, TCP modifications are difficult to deploy, as TCP resides in the system kernel of a device. To relief this issue, Google has recently proposed a new transport layer protocol, called the Quick UDP Internet Connection (QUIC) protocol [23]. Being based on UDP, QUIC can be implemented in the user space rather than the kernel, and can therefore be deployed and updated more easily than TCP. To guarantee reliability, QUIC has to implement a congestion control algorithm, similarly as for TCP. Any kind of congestion control algorithms can be implemented and, therefore, also those already developed for TCP. QUIC provides several interesting improvements compared to TCP. First, 0-RTT connection establishment when client and server have already communicated in the past, which helps reducing latency. Second, true multiplexing of HTTP/2 streams at the transport level, as opposed to standard TCP that still introduces head-of-line blocking when the packets of a certain stream are lost. Third, the possibility

to easily deploy new congestion control algorithms. The Google QUIC team reports a rebuffering rate reduction of YouTube playbacks by 18.0% for desktop users and 15.3% for mobile users [51], when using QUIC compared to standard TCP. Moreover, the number of videos played at their optimal rate increases by 2.9% for desktop and by 4.6% for mobile users, respectively. These results have been obtained by deploying the QUIC protocol globally for Google services. As pointed out by this work, QUIC introduces major innovations that can be useful in HAS, and is especially useful whenever network congestion, loss, and RTT are high. Nevertheless, a formal analysis of adaptive streaming over QUIC has only been marginally investigated [96], and is still missing at the moment. In light of the above, *what is the impact of QUIC on adaptive streaming and in which network conditions does it outperform standard TCP?* Moreover, *is it possible to develop a congestion control algorithm tailored on the adaptive streaming needs?* As seen in Section 3.3, TCP modifications are somewhat limited by the actual applicability of the solution and the a priori limitations of TCP. As listed above, QUIC opens up a new range of possibilities to develop real HAS-aware congestion control algorithms.

5.3 Traffic Encryption

Nowadays, privacy and security have become two of the main requirements for Internet users. Therefore, it is expected that an increasingly larger portion of Internet traffic will be encrypted in the next few years. As an example, QUIC traffic is encrypted by design while HTTP/2 is only supported with encryption enabled by web browsers. This trend concerns video streaming applications as well. Despite being beneficial from a user perspective, traffic encryption can pose a serious challenge on network operations. As seen in Section 3.1, most of the network-assisted solutions rely on the possibility of intercepting and analyzing video traffic to carry out their optimizations. Encryption will make these solutions more and more difficult to apply in future years, unless the streaming provider and the network provider overlap or agree to collaborate. In order to allow network-based solutions to fully optimize HAS streams, QoE factors would need to be estimated. Consequently, *is it possible to infer QoE factors inside the network for encrypted HAS traffic?* Machine learning algorithms will play a very relevant role in this case. For instance, Dimopoulos et al. have shown that it is possible to classify QoE events for YouTube encrypted traffic, in terms of stalls, quality switches and average quality [27]. A similar conclusion is also drawn by Orsolic et al. [71]. We expect a strong research focus in this area, to create prediction algorithms that can classify encrypted traffic online and with high accuracy, two requirements for the applicability of network-assisted solutions. Moreover, the impact of encryption on user experience has not been investigated so far. In other words, *what is the contribution of encryption in video streaming on the users' QoE?* For example, a user might be willing to trade a certain level of encryption in exchange of a superior QoE, which can be provided by using network solutions. In the future, we can expect personalized QoE models including privacy and security together with classical video streaming metrics.

5.4 Personalized QoE-centric control

As reported in Section 3, a large body of research has investigated how to improve the delivery of adaptive video streams, taking QoE parameters explicitly into account. Despite that, these works still suffer from three inefficiencies. First, the employed QoE models are developed to capture the behavior of the "average" user, and are therefore not personalized. Second, standard models do not consider the context in which the streaming session takes place. Third, only the QoE model of the users is inserted into the control loop, but not the user itself. In other words, the actual online user experience is not captured and is not used in the online optimization of the streaming service. Three challenges can be identified in the domain of personalized QoE-centric control. First, *how can a QoE*

model for HAS, which is representative of the specific, rather than aggregate, user behavior be created? Estimating QoE does not only involve application-level parameters, such as switches or freezes, but also sensorial inputs and context [79, 81]. Current smartphones and tablets are already equipped with several sensors (e.g., GPS, light sensors etc.), and wearable devices will allow to obtain even more fine-grained information on the user status (e.g., heart level, eye-tracking etc.). Using both network-, application- and user-level parameters will allow to create real personalized QoE models. Second, *where should the QoE model computation be carried out?* Machine learning algorithms represent a good candidate for the correlation of the aforementioned different parameters. As the input space increases though, it could become impossible to learn such a complex model directly on the user's device. Offloading such computational intensive task to the cloud can represent a viable solution. In this scenario, the real-time delay restrictions of the QoE modeling task should be carefully taken into account. Third, *how can the online feedback of the user be included into the control of the adaptive streaming service?* This feedback can be both explicit, if the user can directly rate the viewing experience, and implicit, when the sensorial information is used to estimate the personalized user's QoE. Particularly, a complex model that encompasses not only application-level but also user-level parameters could be used to estimate on-the-fly and online the real QoE of the specific user. This aspect opens up the possibility to control the service directly at the user level, in order to enforce specific QoE levels that are representative of the real way the user perceives the video streaming session.

5.5 Video Delivery for Low-Latency and High-Mobility Applications

Low-latency streaming applications, as immersive streaming, gaming and real-time communication, will become dominant in future years. Moreover, the current trend for telecommunication networks is to evolve towards large scale deployments that can encompass billions of devices, the so-called Internet of Things (IoT). Many of these devices will have streaming capabilities as well. For instance, as vehicles are becoming smart and connected, in-car entertainment services and applications will gain momentum. Swarms of drones are already being deployed to monitor and support the coverage of live events or emergency operations. All these scenarios can be characterized by a high degree of mobility, low-latency or reliability requirements, or a combination thereof. Even though mobility and latency have already been investigated in the adaptive streaming domain, a shift occurs in the delivery architecture of these new applications. As an example, autonomous cars can both stream from a stable network infrastructure or from each other [39]. In light of the above, *how can continuous, reliable and low-latency video streams be provided in an IoT scenario?* Adaptive video streaming will have a central role to enable these dynamic services, especially when combined with new network paradigms as 5G and softwarized networks [32]. 5G will be able to support both traditional forms of communication and more unstructured ones, as machine-to-machine communication. This aspect entails that the network has to be flexible and able to dynamically reconfigure itself, depending on the application and its requirements. Context-awareness, both applicable in the video client (as reported in Section 3.2.2) and in the network, will also be crucial in this domain. In such a dynamic ecosystem, clients' conditions can drastically change or be extremely unreliable. Being aware of these characteristics will be essential to provide the best service to the final users.

5.6 Open Software and Dataset Availability

In Section 4 we attempted to provide a general overview of the benefits and drawbacks of current QoE-centric solutions for adaptive streaming. One of the main procedural challenges that complicate this task is the lack of a common open-source platform or framework that would allow to compare

different strategies among each other. Some attempts in this direction have already been carried out, for example by De Cicco et al. and Schwarzmann et al. [24, 86]. The former allows to easily compare different rate adaptation heuristics, while the latter allows to compare network-assisted strategies. Stohr et al. implements an emulation environment for the systematical comparison and analysis of different DASH players and rate adaptation heuristics [92]. Moreover, open access to experimentation facilities and testbeds[1] can stimulate research efforts in this direction. Public datasets, in terms of HAS videos [53] or bandwidth conditions [82, 99], are also very important when it comes to the replicability of results. This trend should be encouraged in future years, in order provide a common ground for the development and comparison of new and existing solutions in the adaptive streaming domain.

6 CONCLUSIONS

Optimizing the QoE of HAS users is a complex and challenging task. Purely client-based rate adaptation heuristic might not reach optimal performance, as they only possess a local and decentralized knowledge of the streaming and network conditions. In this survey, we therefore reviewed the *status* of existing works in the adaptive streaming domain that go beyond classical adaptation heuristics. Particularly, we categorized these works in three groups, based on where the optimization takes place. First, server- and network-assisted solutions place additional intelligence in the network to support the delivery of the video. Traffic rerouting, bandwidth shaping techniques and caching represent the most typical examples in this space. This research area can greatly benefit from the new MPEG-SAND standard, which standardizes the possible messages network nodes can exchange to optimize HAS streams. Second, application level solutions can optimize the behavior of any adaptation heuristic by exploiting, for instance, the new features of the HTTP/2 protocol or prefetching techniques. Third, transport level approaches modify the congestion control algorithms or retransmission policy of TCP, to better support video traffic. All these different solutions have also been analyzed in terms of QoE benefits (e.g., quality, freezes, fairness etc.) and deployment complexity. In conclusion, a general *rule-them-all* solution for HAS, able to guarantee great and general benefits in terms of QoE factors and easy deployability, has not been developed yet. Network-based solutions provide the greatest advantages in terms of QoE improvements, as the network actively collaborates and supports the streaming clients. Unfortunately, this aspect also challenges the actual deployability of these approaches. Similar considerations can be repeated for transport level solutions. Application level optimizations are easy to deploy but can only provide consistent improvements on specific QoE factors, for example live latency.

Finally, we also presented the main *challenges* and associated research directions currently emerging in HAS services, namely (1) immersive video streaming, (2) QUIC-based adaptive streaming, (3) online analysis of encrypted video traffic, (4) personalized QoE control, (5) video delivery for low-latency, high-mobility applications and (6) open software and dataset availability. These new challenges will be of central interest for the multimedia community in the coming years.

ACKNOWLEDGMENTS

Jeroen van der Hooft is funded by a grant of the Agency for Innovation by Science and Technology in Flanders (VLAIO). This research was performed partially within the imec ICON PRO-FLOW project (150223).

[1] https://www.fed4fire.eu/

REFERENCES

[1] H. Ahlehagh and S. Dey. 2013. Adaptive Bit Rate Capable Video Caching and Scheduling. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. 1357–1362.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. 2012. A survey of information-centric networking. *IEEE Communications Magazine* 50, 7 (July 2012), 26–36.

[3] A. Ahmad, A. Floris, and L. Atzori. 2016. QoE-centric service delivery: A collaborative approach among OTTs and ISPs. *Computer Networks* 110 (2016), 168 – 179.

[4] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. 2013. Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13)*. ACM, New York, NY, USA, 19–24.

[5] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis. 2012. An Experimental Evaluation of Rate-adaptive Video Players over HTTP. *Image Commun.* 27, 4 (April 2012), 271–287.

[6] K. T. Bagci, K. E. Sahin, and A. M. Tekalp. 2016. Queue-Allocation Optimization for Adaptive Video Streaming over Software Defined Networks with Multiple Service-Levels. In *2016 IEEE International Conference on Image Processing (ICIP)*. 1519–1523.

[7] A. Bentaleb, A. C. Begen, and R. Zimmermann. 2016. SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*. 1296–1305.

[8] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz. 2017. Network Assisted Content Distribution for Adaptive Bitrate Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 62–75.

[9] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig. 2016. Open Connect Everywhere: A Glimpse at the Internet Ecosystem through the Lens of the Netflix CDN. *CoRR* abs/1606.05519 (2016). http://arxiv.org/abs/1606.05519

[10] N. Bouten, M. Claeys, B. Van Poecke, S. Latré, and F. De Turck. 2016. Dynamic Server Selection Strategy for Multi-Server HTTP Adaptive Streaming Services. In *2016 12th International Conference on Network and Service Management (CNSM)*. 82–90.

[11] N. Carlsson, D. Eager, V. Krishnamoorthi, and T. Polishchuk. 2017. Optimized Adaptive Streaming of Multi-Video Stream Bundles. *IEEE Transactions on Multimedia* PP, 99 (2017), 1–1.

[12] P. Casas, P. Fiadino, A. Sackl, and A. D'Alconzo. 2014. YouTube in the move: Understanding the performance of YouTube in cellular networks. In *2014 IFIP Wireless Days (WD)*. 1–6.

[13] C. Cetinkaya, E. Karayer, M. Sayit, and C. Hellge. 2014. SDN for Segment Based Flow Routing of DASH. In *2014 IEEE Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin)*. 74–77.

[14] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. 2013. A Scheduling Framework for Adaptive Video Delivery over Cellular Networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking (MobiCom '13)*. ACM, New York, NY, USA, 389–400.

[15] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori. 2015. DASH Fast Start Using HTTP/2. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '15)*. ACM, New York, NY, USA, 25–30.

[16] Cisco. 2016. Cisco Visual Networking Index: Forecast and Methodology, 2016–2021. http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf. (2016).

[17] M. Claeys, N. Bouten, D. De Vleeschauwer, K. De Schepper, W. Van Leekwijck, S. Latré, and F. De Turck. 2016. Deadline-Aware TCP Congestion Control for Video Streaming Services. In *2016 12th International Conference on Network and Service Management (CNSM)*. 100–108.

[18] M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré, and F. De Turck. 2016. Cooperative Announcement-Based Caching for Video-on-Demand Streaming. *IEEE Transactions on Network and Service Management* 13, 2 (June 2016), 308–321.

[19] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. 2017. Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* (2017).

[20] S. Colonnese, F. Cuomo, R. Guida, and T. Melodia. 2015. Performance Evaluation of Sender-Assisted HTTP-Based Video Streaming in Wireless Ad Hoc Networks. *Ad Hoc Networks* 24, Part B (2015), 74 – 84.

[21] CONVIVA. 2015. Mid-Year 2015 Update: Conviva Viewer Experience Report. http://www.conviva.com/conviva-viewer-experience-report/midyear-vxr-2015/. (2015).

[22] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon. 2016. Cross-layer Scheduler for Video Streaming over MPTCP. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 7, 12 pages.

[23] Y. Cui, T. Li, C. Liu, X. Wang, and M. Kuehlewind. 2017. Innovating Transport with QUIC: Design Approaches and Research Challenges. *IEEE Internet Computing* 21, 2 (Mar 2017), 72–76.

[24] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. 2014. TAPAS: A Tool for rApid Prototyping of Adaptive Streaming Algorithms. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming (VideoNext '14)*. ACM, New York, NY, USA, 1–6.

[25] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. 2013. Model for Estimating QoE of Video Delivered Using HTTP Adaptive Streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. 1288–1293.

[26] R. Deng and G. Liu. 2017. QoE Driven Cross-Layer Scheme for DASH-Based Scalable Video Transmission over LTE. *Multimedia Tools and Applications* (2017), 1–25.

[27] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 513–526.

[28] H. E. Egilmez, S. Civanlar, and A. M. Tekalp. 2013. An Optimization Framework for QoS-Enabled Adaptive Video Streaming Over OpenFlow Networks. *IEEE Transactions on Multimedia* 15, 3 (April 2013), 710–715.

[29] A. E. Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada. 2015. QoE-Based Traffic and Resource Management for Adaptive HTTP Video Delivery in LTE. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 6 (June 2015), 988–1001.

[30] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. 2012. Interactions Between HTTP Adaptive Streaming and TCP. In *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12)*. ACM, New York, NY, USA, 21–26.

[31] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu. 2017. Fixation Prediction for 360&Deg; Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM, New York, NY, USA, 67–72.

[32] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. 2017. Network Slicing in 5G: Survey and Challenges. *IEEE Communications Magazine* 55, 5 (May 2017), 94–100.

[33] B. Fu, D. Staehle, G. Kunzmann, E. Steinbach, and W. Kellerer. 2015. QoE-Based SVC Layer Dropping in LTE Networks Using Content-Aware Layer Priorities. *ACM Trans. Multimedia Comput. Commun. Appl.*, Article 7 (Aug. 2015), 23 pages.

[34] A. Ganjam, J. Jiang, X. Liu, V. Sekar, F. Siddiqi, I. Stoica, J. Zhan, and H. Zhang. 2015. C3: Internet-scale Control Plane for Video Quality Optimization. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)*. USENIX Association, Berkeley, CA, USA, 131–144.

[35] C. Ge, N. Wang, S. Skillman, G. Foster, and Y. Cao. 2016. QoE-Driven DASH Video Caching and Adaptation at 5G Mobile Edge. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking (ACM-ICN '16)*. ACM, New York, NY, USA, 237–242.

[36] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. 2013. Towards Network-Wide QoE Fairness Using Openflow-Assisted Adaptive Video Streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking (FhMN '13)*. ACM, New York, NY, USA, 15–20.

[37] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*. ACM, New York, NY, USA, 129–143.

[38] B. Han, X. Wang, N. Choi, T. Kwon, and Y. Choi. 2013. AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 375–380.

[39] H. He, H. Shan, A. Huang, and L. Sun. 2016. Resource Allocation for Video Streaming in Heterogeneous Cognitive Vehicular Networks. *IEEE Transactions on Vehicular Technology* 65, 10 (Oct 2016), 7917–7930.

[40] T. Hossfeld, M. Seufert, C. Sieber, and T. Zinner. 2014. Assessing Effect Sizes of Influence Factors Towards a QoE Model for HTTP Adaptive Streaming. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. 111–116.

[41] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*. ACM, New York, NY, USA, 187–198.

[42] R. Huysegems, J. van der Hooft, T. Bostoen, P. Rondao Alface, S. Petrangeli, T. Wauters, and F. De Turck. 2015. HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 541–550.

[43] ISO/IEC 23009-5:2017 2017. *Information Technology – Dynamic Adaptive Streaming over HTTP (DASH) – Part 5: Server and Network Assisted DASH (SAND).* Standard. International Organization for Standardization.

[44] C. James, E. Halepovic, M. Wang, R. Jana, and N. K. Shankaranarayanan. 2016. Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH?. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 331–336.

[45] Y. Jin, Y. Wen, and C. Westphal. 2015. Optimal Transcoding and Caching for Adaptive Streaming in Media Cloud: an Analytical Approach. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 12 (Dec 2015), 1914–1925.

[46] J. W. Kleinrouweler, S. Cabrero, and P. Cesar. 2016. Delivering Stable High-Quality Video: An SDN Architecture with DASH Assisting Network Elements. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 4, 10 pages.

[47] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri. 2015. Bandwidth-Aware Prefetching for Proactive Multi-Video Preloading and Improved HAS Performance. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 551–560.

[48] D. K. Krishnappa, M. Zink, C. Griwodz, and P. Halvorsen. 2015. Cache-Centric Video Recommendation: An Approach to Improve the Efficiency of YouTube Caches. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 4, Article 48 (June 2015), 20 pages.

[49] D. K. Krishnappa, M. Zink, and R. K. Sitaraman. 2015. Optimizing the Video Transcoding Workflow in Content Delivery Networks. In *Proceedings of the 6th ACM Multimedia Systems Conference (MMSys '15)*. ACM, 37–48.

[50] J. Kua, G. Armitage, and P. Branch. 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP. *IEEE Communications Surveys Tutorials* PP, 99 (2017), 1–1.

[51] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 183–196.

[52] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner. 2013. An Experimental Analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6.

[53] S. Lederer, C. Müller, and C. Timmerer. 2012. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*. ACM, New York, NY, USA, 89–94.

[54] D. H. Lee, C. Dovrolis, and A. C. Begen. 2014. Caching in HTTP Adaptive Streaming: Friend or Foe?. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV '14)*. ACM, New York, NY, USA, Article 31, 6 pages.

[55] C. Li, P. Frossard, H. Xiong, and J. Zou. 2016. Distributed Wireless Video Caching Placement for Dynamic Adaptive Streaming. In *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '16)*. ACM, New York, NY, USA, Article 2, 6 pages.

[56] W. Li, S. Oteafy, and H. Hassanein. 2017. Rate-Selective Caching for Adaptive Streaming over Information-centric Networks. *IEEE Trans. Comput.* PP, 99 (2017), 1–1.

[57] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. 2014. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (April 2014), 719–733.

[58] K. Liang, J. Hao, R. Zimmermann, and D. K. Y. Yau. 2015. Integrated Prefetching and Caching for Adaptive Video Streaming over HTTP: An Online Approach. In *Proceedings of the 6th ACM Multimedia Systems Conference (MMSys '15)*. ACM, New York, NY, USA, 142–152.

[59] E. Liotou, T. Hossfeld, C. Moldovan, F. Metzger, D. Tsolkas, and N. Passas. 2016. Enriching HTTP Adaptive Streaming with Context Awareness: A Tunnel Case Study. In *2016 IEEE International Conference on Communications (ICC)*. 1–6.

[60] Y. Liu, J. Geurts, J. C. Point, S. Lederer, B. Rainer, C. Mueller, C. Timmerer, and H. Hellwagner. 2013. Dynamic Adaptive Streaming over CCN: A Caching and Overhead Analysis. In *2013 IEEE International Conference on Communications (ICC)*. 3629–3633.

[61] Z. Lu, V. S. Somayazulu, and H. Moustafa. 2014. Context-Adaptive Cross-layer TCP Optimization for Internet Video Streaming. In *2014 IEEE International Conference on Communications (ICC)*. 1723–1728.

[62] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu. 2017. Understanding Performance of Edge Content Caching for Mobile Video Streaming. *IEEE Journal on Selected Areas in Communications* 35, 5 (May 2017), 1076–1089.

[63] A. Mansy, M. Fayed, and M. Ammar. 2015. Network-Layer Fairness for Adaptive Video Streams. In *2015 IFIP Networking Conference (IFIP Networking)*. 1–9.

[64] S. McQuistin, C. Perkins, and M. Fayed. 2016. TCP Goes to Hollywood. In *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '16)*. ACM, New York, NY, USA, Article 5, 6 pages.

[65] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. 2012. QDASH: A QoE-Aware DASH System. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*. ACM, 11–22.

[66] C. Mueller, S. Lederer, C. Timmerer, and H. Hellwagner. 2013. Dynamic Adaptive Streaming over HTTP/2.0. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6.

[67] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. 2015. Practical, Real-time Centralized Control for CDN-based Live Video Delivery. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 311–324.

[68] H. Nam, K. H. Kim, J. Y. Kim, and H. Schulzrinne. 2014. Towards QoE-Aware Video Streaming using SDN. In *2014 IEEE Global Communications Conference*. 1317–1322.

[69] D. V. Nguyen, H. T. Le, P. N. Nam, A. T. Pham, and T. C. Thang. 2016. Request Adaptation for Adaptive Streaming over HTTP/2. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*. 189–191.

[70] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim. 2016. MPEG DASH SRD: Spatial Relationship Description. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 5, 8 pages.

[71] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. 2017. A machine learning approach to classifying YouTube QoE based on encrypted network traffic. *Multimedia Tools and Applications* (06 May 2017).

[72] K. Park, I. Bouazizi, and Y. Xu. 2017. Use cases and draft requirements for Network Based Media Processing. https://mpeg.chiariglione.org/standards/exploration/network-based-media-processing/requirements-network-based-media-processing-v1. (2017).

[73] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages.

[74] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360° Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*. ACM, New York, NY, USA, 306–314.

[75] S. Petrangeli, T. Wu, T. Wauters, R. Huysegems, T. Bostoen, and F. D. Turck. 2017. A machine learning-based framework for preventing video freezes in HTTP adaptive streaming. *Journal of Network and Computer Applications* 94, Supplement C (2017), 78 – 92.

[76] W. Pu, Z. Zou, and C. W. Chen. 2012. Video Adaptation Proxy for Wireless Dynamic Adaptive Streaming over HTTP. In *2012 19th International Packet Video Workshop (PV)*. 65–70.

[77] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. 2016. Optimizing 360 Video Delivery over Cellular Networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC '16)*. ACM, New York, NY, USA, 1–6.

[78] B. Rainer, D. Posch, and H. Hellwagner. 2016. Investigating the Performance of Pull-Based Dynamic Adaptive Streaming in NDN. *IEEE Journal on Selected Areas in Communications* 34, 8 (Aug 2016), 2130–2140.

[79] B. Rainer and C. Timmerer. 2014. A Generic Utility Model Representing the Quality of Sensory Experience. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 1s, Article 14 (Oct. 2014), 17 pages.

[80] A. Ramakrishnan, C. Westphal, and J. Saltarin. 2016. Adaptive Video Streaming over CCN with Network Coding for Seamless Mobility. In *2016 IEEE International Symposium on Multimedia (ISM)*. 238–242.

[81] P. Reichl, S. Egger, S. Moeller, K. Kilkki, M. Fiedler, T. Hossfeld, C. Tsiaras, and A. Asrese. 2015. Towards a Comprehensive Framework for QoE and User Behavior Modelling. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. 1–6.

[82] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. 2012. Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning. *ACM Trans. Multimedia Comput. Commun. Appl.* 8, 3, Article 24 (Aug. 2012), 19 pages.

[83] J. Ryoo, K. Yun, D. Samaras, S. R. Das, and G. Zelinsky. 2016. Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 6, 11 pages.

[84] Y. Sani, A. Mauthe, and C. Edwards. 2017. Adaptive Bitrate Selection: A Survey. *IEEE Communications Surveys Tutorials* PP, 99 (2017), 1–1.

[85] H. Schwarz, D. Marpe, and T. Wiegand. 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9 (Sept 2007), 1103–1120.

[86] S. Schwarzmann, T. Zinner, and O. Dobrijevic. 2016. Towards a Framework for Comparing Application-Network Interaction Mechanisms. In *2016 28th International Teletraffic Congress (ITC 28)*, Vol. 03. 13–18.

[87] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia. 2014. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials 99, 2014* (2014).

[88] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer. 2015. The cost of aggressive HTTP adaptive streaming: Quantifying YouTube's redundant traffic. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 1261–1267.

[89] I. Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia* 18, 4 (April 2011), 62–67.

[90] M. Song, Y. Lee, and J. Park. 2015. Scheduling a Video Transcoding Server to Save Energy. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 2s, Article 45 (Feb. 2015), 23 pages.

[91] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.

[92] D. Stohr, A. Frömmgen, A. Rizk, M. Zink, R. Steinmetz, and W. Effelsberg. 2017. Where Are the Sweet Spots?: A Systematic Approach to Reproducible DASH Player Comparisons. In *Proceedings of the 2017 ACM on Multimedia Conference (MM '17)*. ACM, New York, NY, USA, 1113–1121.

[93] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli. 2016. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference (SIGCOMM '16)*. ACM, New York, NY, USA, 272–285.

[94] S. Tavakoli, S. Egger, M. Seufert, R. Schatz, K. Brunnström, and N. García. 2016. Perceptual Quality of HTTP Adaptive Streaming Strategies: Cross-Experimental Analysis of Multi-Laboratory and Crowdsourced Subjective Studies. *IEEE Journal on Selected Areas in Communications* 34, 8 (Aug 2016), 2141–2153.

[95] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey. 2017. Enhancing MPEG DASH Performance via Server and Network Assistance. *SMPTE Motion Imaging Journal* 126, 1 (Jan 2017), 22–27.

[96] C. Timmerer and A. Bertoni. 2016. Advanced Transport Options for the Dynamic Adaptive Streaming over HTTP. *CoRR* abs/1606.00264 (2016). http://arxiv.org/abs/1606.00264

[97] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard. 2015. Optimal Selection of Adaptive Streaming Representations. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 2s, Article 43 (Feb. 2015), 26 pages.

[98] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck. 2015. A Learning-Based Algorithm for Improved bandwidth-Awareness of Adaptive Streaming Clients. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 131–138.

[99] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (Nov 2016), 2177–2180.

[100] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck. 2017. An HTTP/2 Push-Based Approach for Low-Latency Live Streaming with Super-Short Segments. *Journal of Network and Systems Management* (2017), 1–28.

[101] C. Wang, H. Kim, and R. Morla. 2015. QoE Driven Server Selection for VoD in the Cloud. In *2015 IEEE 8th International Conference on Cloud Computing*. 917–924.

[102] C. Wang, A. Rizk, and M. Zink. 2016. SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 1, 12 pages.

[103] Z. Wang, L. Sun, C. Wu, W. Zhu, and S. Yang. 2014. Joint Online Transcoding and Geo-Distributed Delivery for Dynamic Adaptive Streaming. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 91–99.

[104] S. Wei and V. Swaminathan. 2014. Low Latency Live Video Streaming over HTTP 2.0. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV '14)*. ACM, New York, NY, USA, Article 37, 6 pages.

[105] S. Wei, V. Swaminathan, and M. Xiao. 2015. Power Efficient Mobile Video Streaming using HTTP/2 Server Push. In *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*. 1–6.

[106] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (July 2003), 560–576.

[107] M. Xiao, V. Swaminathan, S. Wei, and S. Chen. 2016. DASH2M: Exploring HTTP/2 for Internet Streaming to Mobile Devices. In *Proceedings of the 2016 ACM on Multimedia Conference (MM '16)*. ACM, New York, NY, USA, 22–31.

[108] Y.-T. Yu, F. Bronzino, R. Fan, C. Westphal, and M. Gerla. 2015. Congestion-Aware Edge Caching for Adaptive Video Streaming in Information-Centric Networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. 588–596.

[109] M. Zhao, X. Gong, J. Liang, W. Wang, X. Que, and S. Cheng. 2015. QoE-Driven Cross-Layer Optimization for Wireless Dynamic Adaptive Streaming of Scalable Videos Over HTTP. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 3 (March 2015), 451–465.

[110] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer. 2014. Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects and Challenges. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. 1–6.

Received June 2017; revised 0000 0000; accepted 0000 0000